System Services CSCI

Network Services CSC

Thor

Design Panel 2/3 Review

November 18, 1997

Network Services CSC

1.1 Network Services Introduction

1.1.1 Network Services Overview

The Network Services CSC for Thor will include the following:

- Basic Communication Service
- Network APIs
- Activity Separation

These services are resident on the RTPS platforms as outlined below.

1.1.2 Network Services Operational Description

The Network Services provides the capability by which end system applications, services and users communicate and distribute data within CLCS. The three major elements of the Network Services are: Basic Communication Services, Network APIs, and Activities & Activity Separation.

1.1.2.1 Basic Communication Services:

The Basic Communications Services, listed below, consist of LAN services and protocols which are supplied by the COTS OS.

- Transmission Control Protocol/ Internet Protocol (TCP/IP): TCP/IP is provided as a COTS product.
- File Transfer Protocol (FTP): The FTP protocol is provided as a COTS product.
- Telephone Network (TELNET): Telnet is provided as a COTS product.
- Network Information Service (NIS): The NIS will be implemented by a NIS server platform that is resident in each Flow Zone. This server will have direct connectivity to each Operational network (RTCN, DCN) within its associated Flow Zone and any attached Control and Gateway Groups. It will provide password control for users utilizing the resources within the string of equipment.
- Network File System (NFS): The Network File System is provided as a COTS product. It's configuration is controlled by the Operating Systems Group.
- Network Timing Protocol **Error! Bookmark not defined.** (ntp): The Network Timing Protocol provides a mechanism to synchronize the clocks residing on local hosts with both a common timing source and each other. The common timing source will be a Datum timing box that accepts an IRIG-B timing signal from Timing and Countdown (T&CD) and functions as an ntp server. This Datum box will be present in each Flow Zone and act as the time reference for all local **ntp servers** contained within that Flow Zone as well as any Control Groups that may be attached to that Flow Zone. **For Thor, a hierarchical implementation will be considered. Multiple hosts would be selected to synchronize their time directly from the Datum box. The balance of the hosts within a set would then synchronize to those multiple hosts, providing a level of redundancy for ntp. (The choice of hosts to act as ntp servers is an open issue. After the Design Panel discussions, an ERP issue will be opened concerning this subject.)**
- UNIX r-commands (e.g., rlogin): The Unix r-commands are provided as a COTS product.

1.1.2.2 Network APIs:

The Network APIs consist of a common set of custom developed library calls which applications can use for data distribution.

Application Messaging consists of a set of APIs for Connection-Oriented Point-To-Point communications. Client Applications will be able to establish and open a connection with applications on a remote server machine. Once the connection is established, data can be sent/received to/from the remote machine.
Connectionless Messaging consists of a set of APIs for Connectionless Point-To-Point communications. Local applications will use CLM for sending datagrams Point-To-Point to a specific host machine - no confirmation for transmission or acknowledge of delivery is required from the receiver. The CLM API will also support the transmission of Reliable Multicast data - one transmitter to a group of receivers.
Network Registration Service (NRS). A method for determining the IP assignment of file descriptors, the location of applications and the distribution of this information is required in order for applications to send and receive data on the network. Static registration files or tables will be the mechanism used on the RTCN and for multicast address resolution . The registration files or tables will be maintained through manual procedures. The NRS will be utilized on the DCN for point-to-point communications. NRS allows applications to register their existence by hostname and port number. Other applications can then locate these applications through the NRS APIs. This information is broadcasted by each local NRS process onto the network whereby, the NRS process on the other machines update their local NRS information.

1.1.2.3 Activities & Activity Separation:

Activity will be a term utilized to classify an operation in the CLCS. A Naming Standard for data streams will be developed which uses parameters from the Activity definition (i.e., TCID, SCID, Vehicle or Tail Number, Live Vehicle or Sim, etc.). Logical Separation can then be made by allowing applications in one Activity to only communicate on their Activity.

Error! Bookmark not defined.

An activity will be defined via the OPS CM Activity Manager. When a set of hardware (Gateway Group(s), Control Group(s), Flow Zone(s)) is assigned to support a particular activity, the CM Server will download the activity load to the specific HWCI's supporting that activity utilizing an Activity Manager function. As a part of the load procedure, the host tables for each HWCI will be manipulated (utilizing NRS) in such a manner that only those HWCI's that constitute that particular activity will have visibility to only those HWCI's within their activity. HWCI's that provide services common across activities (ie: NIS servers, etc_) will maintain their capability to see (and be seen by) all other platforms.

1.2 Network Services Specifications

1.2.1 Network Services Groundrules

- The Network Services API will be designed with the following RTPS LAN requirements into consideration:

  A) The RTPS LAN will be fault tolerant, such that, a single LAN component will not cause more than one end station failure.
  B) The switching of a LAN component to a backup (due to a failure) will be deterministic and within the Reliable Message time allocation.
  C) Fault isolation and LAN component replacement will be conducted with minimum interference to the on-going operation.
  D) An industry LAN standard protocol for supporting one-to-many communications will be utilized, such that, interoperability among different LAN products can be obtained.

- **As a part of Redstone and early Thor activities, studies were conducted to examine the current Reliable Messaging implementation. Specifically, the current Library-based implementation was compared to a Process-based implementation as well as comparing the ACK based protocol to multiple protocols (dependent upon the type of traffic being generated) utilizing ACKs or NACKs. The study results were presented to the ERP and a total rewrite of RM was approved. This rewrite, which is obviously a major effort, will be performed in parallel with the Thor effort, will go through a complete Design Panel cycle, and will be delivered in the early Atlas timeframe. With this in mind, any new Thor requirements that involve a major change to the current code will be postponed until this Atlas delivery. These requirements are identified in italics.**

1.2.2 Network Services Functional Requirements

NOTE 1: Requirements in this document in *Italics* format are Post Thor and are specified for design consideration and future delivery capabilities (not to be tested in Thor).

The Network Services Functional Requirements area is composed of the following CSC functions:

1. Network APIs
2. Network Registration Services API
3. Activity Separation (Data)
4. Activity Separation (Platform)
5. Basic Communications

1.2.2.1 Network APIs

1. Network Services will provide applications with a common API for communicating across the network interfaces. **(Complete, tested in Redstone)**
2. The Network Services API will provide independent transmit and receive functionality. **(Complete, tested in Redstone)**
3. The API will be capable of reliably sending multicast and broadcast data streams. **(SGI Complete & tested in redstone; VxWorks incomplete, scheduled for Thor, Issue #83)**
4. The API will be capable of supporting connection oriented point-to-point data, connectionless point-to-point data, and connectionless point-to-multipoint data. **(Complete, tested in Redstone)**
5. The Network Services API will provide a mechanism to associate applications with Service Points (SP's). **(Complete, tested in Redstone)**
6. The API will support the fragmentation and reassembly of messages that exceed the physical layer Maximum Transmission Unit (MTU). **(Complete, tested in Redstone)**
7. Data messages will be re-transmitted to end stations registered for reliable message communication participation **(Complete, tested in Redstone)**
8. Re-transmission of the current data message will complete or time-out prior to transmission of the next message received from the sending application. **(Complete, SGI tested in Redstone, VxWorks to be tested in Thor, Issue #82)**
9. At a minimum, failed attempts and repeated retries will be reported to the calling application for transfer to System Messages. **(Complete, tested in Redstone)**
10. The number of data message transmission retries will be statically configurable per data stream type. (Note: Entries will be procedurally controlled to prevent exceeding the system data transmission time limitations) **(Complete, SGI tested in Redstone, VxWorks to be tested in Thor, Issue #81)**
11. Message delivery attempts by the sender will be aborted based on a configurable expiration timer set for acknowledgment responses. **(Complete, SGI tested in Redstone, VxWorks to be tested in Thor, Issue #80)**
12. The API will provide automatic sequential numbering of data packets based on stream or port connections. **(Complete, tested in Redstone)**
13. The API will contain the capability to receive and buffer the next application message(s) until the current message is transmitted, re-transmitted or aborted. **(Complete, tested in Redstone)**
14. The receiver will discard the whole message if an error is detected or message re-assembly is unsuccessful. **(Complete, tested in Redstone)**
15. The receiver will report receive errors to the calling application for transfer to System Messages. **(Complete, tested in Redstone)**
16. *In cases where the source of the stream has moved from one platform to another or has closed a stream and then re-opened the same stream, the API will allow receiving applications to continue to receive a data stream without having to register again.* **(Incomplete, to be implemented in Atlas, Issue #79)**
17. Reliable message for connection oriented (point-to-point) communication will be provided as specified by the TCP layer standards. **(Complete, tested in Redstone)**
18. The sender will transmit the multicast and broadcast data even though a client has not yet registered to participate in reliable messaging communication. **(Complete, tested in Redstone)**

19. The API will associate the stream name (defined in the Naming Standard) with a unique multicast address. The stream name assignment to multicast addresses will consist of a static file for Redstone. **(Complete, tested in Redstone)**
20. The API will allow applications to open multiple streams within a single process. **(SGI Complete & tested in redstone; VxWorks incomplete, scheduled for Thor, Issue #78)**
21. The API will provide a function which relieves applications from having to poll for received data. **(SGI Complete & tested in redstone; VxWorks incomplete, scheduled for Thor, Issue #77)**
22. The API software will be structured and designed such that a single software baseline can be obtained regardless of platform type (i.e. VxWorks or UNIX). **(Complete, tested in Redstone)**
23. ***The multicast acknowledgments will be transferred to SDC for recording.*** **(Incomplete, to be implemented in Atlas, Issue #66)**
24. ***The API will contain the ability to enable/disable the transfer of acknowledgments to SDC for recording, with enable as the default setting.*** **(Incomplete, to be implemented in Atlas, Issue #66)**
25. ***The API will provide the capability to support multiple senders on a single stream***. **(Incomplete, to be implemented in Atlas)**

1.2.2.2 Network Registration Service API:

1. The Network Registration Service API will provide the following functions:

    a) A capability that allows applications to add an entry to the registration file.
    b) A locate function that allows applications to locate a registered file entry.
    c) A de-register function that allows applications to remove a registered entry.

2. Registration information will be distributed to all the pertinent platforms assigned to a specific Test Set.
3. All transactions which alter host tables will be logged.
4. Dynamic update of host tables will be provided based on the configured activity on each platform.
5. In point -to-point communications the Network Registration Service will dynamically allocate the port for a given application.
6. Registration of platform name only (without port name) will be permitted.
7. The association of platform and activities will be recorded.
8. The association of platform and activities will be provided through an API.
9. Deactivation of an activity will not interfere with activity resources in use.
10. The following API calls will be used to support registration functions:

    a) register
    b) get port
    c) port register
    d) de-register
    e) search
    f) port search
    g) platform list
    h) activate activity
    i) de-activate activity
    j) get activity
    k) get activity list

11. The Network Registration Service will log all API requests and major network events.
12. The Network Registration Service will provide the capability to register separate point-to-point service IDs by activity.

1.2.2.3 Activity Separation (Data)

1. A naming convention will be provided which maps or is associated to a unique multicast data stream.
2. Receivers will be able to only input the data streams associated with its selected activity.

1.2.2.4 Activity Separation (Platform):

1. Logical separation of platforms will be maintained on the network between different types of activities.
2. A capability will be provided to keep platforms supporting one activity type invisible to platforms supporting another activity.
3. The data separation function will allow certain positions to communicate with devices regardless of the activity selected. Note: These positions are said to be "Global".
4. Separation of point-to-point data will utilize the platform separation.
5. The activity-independent or Global workstation and servers will be visible to all workstations at all times.
6. Uncommitted workstations will not have access to workstations and servers committed to an activity.
7. Each workstation will have access to all workstations and servers which are part of the same activity.

1.2.2.5 Basic Communication Services:

1. The Basic Communication Services will provide the functionality of the following services as defined by the COTS TCP/IP Standards:

Service	Gateway	DDP	CCP	HCI
1.2.2.5.1.1 File Transfer Protocol (ftp)	Yes	Yes	Yes	Yes
1.2.2.5.1.2 Telnet	Yes	Yes	Yes	Yes
1.2.2.5.1.3Unix r-commands	No	Yes	Yes	Yes
1.2.2.5.1.4 Network File System (NFS)	No	Yes	Yes	Yes
1.2.2.5.1.5 Network Timing Protocol (ntp)	No	Yes	Yes	Yes
1.2.2.5.1.6 Network Information Service (NIS)	No	Yes	Yes	Yes

2. A command line interface will be provided for the ftp, Telnet, and Unix r-command services.
3. *The Basic Communication Services function will support the Simple Network Management Protocol (SNMP).*
4. Network Services will provide logical communications such that applications on the same platform can communicate transparently as if they were remote on the network. This is in support of CLCS Application Groupings or Location Transparency onto a single platform

1.2.3 Network Services Performance Requirements

Note: Many of these performance requirements are based upon performing certain actions within a specified timeframe. These same requirements span multiple CSCI boundaries. No budget allocations between these CSCIs have been made. These requirements are, therefore, impossible to test at this time in a meaningful way. System Engineering will need to provide these allocations in the very near future if the performance requirements are to be met in the Thor delivery.

1. The Network Registration Service will allow remote de-registration of a specified service identifier (ID) in less than two seconds.
2. The Network Registration Service will propagate new registrations in less than two seconds.
3. The system shall support 25,000 End-Item Function Designator changes per second continuously.

4. The system shall support a peak of 50,000 End-Item Function Designator changes in a given second without losing any data.
5. The system shall support 1,000 End-Item Function Designator changes during a 10 millisecond period.
6. GSE command/response latency of a priority command, or of a non-priority command in an unloaded system, shall be less than 20 milliseconds from the time a test/control application issues the command until the response is received by the test application.
7. GSE command/response latency of a non-priority command in a system supporting the "system maximum data bandwidth" and with 20 test applications executing in the same subsystem shall be less than 100 milliseconds from the time the test/control application issues the command until the response is received by the test application
8. The system shall support executing a manual command in less than one second from a human execution to RTPS interface output.
9. The system shall support (while executing at 40 percent of the system maximum data bandwidth) 25 Constraint Management requests from each of 4 Command and Control Processor Subsystems (100 total), 20 constraint event notifications (5 per CCP), and 4 commands (1 per CCP) with each CCP executing 20 User Test Applications which are executing 500 Application Service calls per second.
10. The RTPS shall provide changed measurement data to system and user applications at the System Synchronous Rate.
11. The RTPS shall provide changed measurement data to display applications at the Display Synchronous Rate.

1.2.4 Network Services Interfaces Data Flow Diagrams

Error! Bookmark not defined.

1.3 Network Services Design Specification

1.3.1 Network Services Detailed Data Flow

1.3.2 Network Services External Interfaces

1.3.2.1 Network Services Message Formats

Since this CSC is strictly a set of library calls provided for the use of calling applications, no System Messages will be generated by the Network Services CSC. Rather, error codes will be generated and passed to the calling applications. The rationale for this implementation is that the overhead required to set up a System Message connection and to generate packets for transmission is beyond the scope of a library call.

```
Enoerr = 0,       /*  0 Error 0 */
Einvalid,         /*  1 invalid parameter */
Eopenclntsoc,     /*  2 open client socket failed */
Econnect,         /*  3 connect(2) to server failed */
Econntimeout,     /*  4 connect timed out */
Eopenservsoc,     /*  5 open server socket failed */
Elisten,          /*  6 listen failed */
EUDPrdwldblk,     /*  7 UDP read of next packet in this message would block */
Esendto,          /*  8 UDP sendto failed */
Eservbind,        /*  9 bind(2) of the socket failed */
Eaccept,          /* 10 accept(2) failed */
Eread,            /* 11 read(2) failed */
Erdbufsmall,      /* 12 incoming message is larger than the receive buffer */
Ewrite,           /* 13 write(2) failed */
Ewritev,          /* 14 writev(2) failed */
Econnwldblk,      /* 15 connect call would block */
Erecvfrom,        /* 16 UDP recvfrom failed */
Eservnotfound,    /* 17 service not found */
Enodenotfound,    /* 18 node not found */
EMCservice,       /* 19 Multicast service not found in IP MC table */
EMCtable,         /* 20 Could not open IP Multicast table */
Eiddselect,       /* 21 select(2) failed in idd_select */
ETCPrdwldblk,     /* 22 TCP read would block */
ETCPwrtwldblk,    /* 23 TCP write would block */
Emalloc,          /* 24 malloc of send buffer space failed */
Epipe,            /* 25 write on a socket with no one to read it */
Eeatmessage,      /* 26 receive buffer too small and attempt to discard extra bytes failed */
Esetrcvbuf,       /* 27 attempt to increase socket receive buffer size failed */
Esetsndbuf,       /* 28 attempt to increase socket send buffer size failed */
Enewsock,         /* 29 attempt to open new socket for UDP segmenting failed */
Esendnewsoc,      /* 30 attempt to send new socket address for segmenting failed */
Ewrongpacket,     /* 31 packet received was not part of message currently being assembled */
Erecvsocwldblk,   /* 32 segmenting UDP message and receive of new server socket timed out */
Erecvnewsoc,      /* 33 attempt to receive new socket address for segmenting failed */
Eclntbind,        /* 34 bind(2) of UNIX client name failed */
EUDPselect,       /* 35 using select(2) to delay between UDP segments failed */
Eaccptwldblk,     /* 36 accept(2) would block */
EUDPsendwldblk,   /* 37 UDP sendto(2) would block */
EUDPrecvwldblk,   /* 38 UDP recvfrom(2) would block */
Eselectintr,      /* 39 select(2) was interrupted by a signal */
ENRSsearch,       /* 40 NRS search for a host for this service failed */
EMCopen,          /* 41 open of IP multicast failed */
EMCioctl,         /* 42 ioctl(2) for LAN device failed */
Eclose,           /* 43 close(2) of socket descriptor failed */
Eunlink,          /* 44 unlink(2) of Unix socket name failed */
Einprogress,      /* 45 The connection is requested on a socket with FNDELAY set */
```

```
Elostbytes,      /* 46 client disconnected before sending all expected bytes */
EBCsocket,       /* 47 socket(2) for broadcast failed */
EBCenable,       /* 48 setsockopt(2) to enable broadcast failed */
Etmpsock,        /* 49 socket(2) for temp socket failed */
Esockrange,      /* 50 socket descriptor is greater than AM/CLM MAX_OPEN_SOCKETS */
EInvalidIPMC,    /* 51 Conversion of IP multicast address to network address failed */
ENoIPMCTable,    /* 52 IP MUlticast table was not found */
EReuseAddr,      /* 53 setsockopt(2) for REUSEADDR failed */
EIntfNotFound,   /* 54 LAN interface not found */
EReUsePort,      /* 55 setsockopt(2) to enable REUSEPORT failed */
ESetMCaddr,      /* 56 setsockopt(2) to enable IP MC address failed */
EIP_MCgroup,     /* 57 setsockopt(2) to join or leave a multicast group failed */
ENOTSND,         /* 58 attempt to send on RM receive stream */
ENOTRCV,         /* 59 attempt to receive on RM send stream */
ETIMEOUT,        /* 60 timeout from one or more RM receivers */
EREXMIT,         /* 61 RM retransmission limit reached for one or more streams */
EBADRSP,         /* 62 unknown RM response opcode */
EMAXRCV,         /* 63 max number of RM receivers reached */
ERMBUG,          /* 64 RM bug encountered -- report to CLCS development */
EDATALOSS,       /* 65 non-fatal data loss occurred -- continuing */
EBADSEQ          /* 66 bad sequence number received, beyond acceptable data loss */
```

1.3.2.2 Network Services Display Formats

This section is not applicable to the Network Services CSC

1.3.2.3 Network Services Input Formats

This section is not applicable to the Network Services CSC.

1.3.2.4 Recorded Data

This section is not applicable to the Network Services CSC.

1.3.2.5 Network Services Printer Formats

This section is not applicable to the Network Services CSC.

1.3.2.6 Interprocess Communications (C-to-C Communications?)

This section is not applicable to the Network Services CSC.

1.3.2.7 Network Services External Interface Calls (e.g., API Calling Formats)

The Network Services API is fully defined in the **Network Services Interface Definition Document (84K00354)**. Provided below is a complete list of the available calls with their parameters. For a full definition, refer to the IDD.

AM/CLM Calls:

- am_open(node, service, type)
- am_connect(sd)
- am_accept(sd)
- am_close(sd)
- am_recv(sd, bufptr, bufsize, log)
- am_send(sd, bufptr, nbytes, log)
- am_get_client(sd)
- am_set_timeout(t)

- clm_open(node, service, type)
- clm_close(sd)
- clm_recv(sd, bufptr, bufsize, log)
- clm_send(sd, msgptr, msgsize, log)
- clm_get_addr(sd)
- clm_set_addr(sd, sp)
- idd_set_bufsize(sd, type, rsize, ssize)
- idd_select(sd, sel, timeout)

NRS Calls:

- nrs_register(service_id)
- nrs_get_port(service_id, port_id)
- nrs_port_reg(service_id, port_id)
- nrs_deregister(service_id)
- nrs_server_deregister(service_id)
- nrs_search(service_id, host_ids)
- nrs_port_srch(service_id, host_ids, port_ids)
- nrs_node_list(node_ids, activity_ids)
- nrs_activate_act(activity)
- nrs_deactivate_act(activity_id)
- nrs_get_act(index, activity)
- nrs_get_act_table(activity)
- nrs_ws_config_act(activity_id)
- NRS_aix_gethostname() **(obsolete)**
- nrs_remote_port_reg(service_id, host_id, act, port_id)
- nrs_remote_cmd(service_id, host_id, cmd_type, act)
- nrs_service_name_srch(service_name, host_ids, serv_ids, port_ids)
- nrs_act_type_node_srch(option, host_ids, act_types, activity_ids)

1.3.2.8 Network Services Table Formats

The only table generated from a source external to the Network Services CSC is the Static Address Table. Each communications path within the system is named in this file and a mapping from this name to themulticast address/port numbers for data and acknowledgments associated with this path is provided. The proposed format for each entry is as follows:

<stream_name> <data_ip_address> <data_udp_port> <timeout_ms> <rexmit_count>

where:
stream_name is the name of the data stream (this format is TBD),
data_ip_address is the address of the data stream (broadcast or multicast),
data_udp_port is the UDP port number associated with the data stream,
timeout_ms is a valid timeout value in milliseconds,
and **rexmit_count is the integer number of times a packet may be sent/resent after timeouts**.

This table is currently generated by hand. It must be present on all hosts that utilize the Reliable Messages protocol. This static address file format allows us to define, at initialization time, whether ack's will be point-to-point or multicast, on a per stream basis. This should prove useful when deciding whether or not to record each stream, and for performance testing.

1.3.3 Network Services Test Plan

A detailed test procedure will be developed that will address the particular network services requirements of the Thor delivery. The Hardware and software requirements are displayed in the following diagram.

1.3.3.1 Test Environment

The #2 Satellite Development Environment (SDE2) facility will be used to test the network services. All processor platforms and network switches that will be required are available as part of the SDE. Each system involved shall be loaded with a baseline image of the Operating System, the network API's, and any test scripts to be executed. In addition, the network elements (i.e. switches) will be loaded with a TBD configuration prior to testing.

The SDE equipment actually participating in the test shall have all network connections removed or disabled except those that are part of the test. This is to verify that the communication is taking place over the CLCS physical networks and not over the currently installed Ethernet.

1.3.3.2 Test Tools

In addition to the working environment of equipment where network services reside, up to two LAN analyzers may also be required for some test procedures. These analyzers will need the ability to capture individual network frames or cells as well as to re-assemble these units into packets from higher level COTS protocols.

1.3.3.3 Test Cases

1.3.3.3.1 Network APIs: Network API testing will be accomplished via the development of simple code sequences that will exercise the AM/CLM calls and NRS calls. These procedures may be run with a network analyzer attached to verify proper retry and timeout thresholds are being upheld, and that packet formats and framing are as expected.

1.3.3.3.2 Network Timing Protocol (ntp): Testing of the ntp implementation will be accomplished by configuring a string of equipment, initiating ntp, and verifying that the multiple server architecture provides redundancy for the local hosts.

APPENDIX A

Statement of Work

- Implement study phase finding from Redstone.
- Maintain statistics on packet rates, data rates, **and** errors. ~~and CPU utilization~~.
- Provide performance data for system modeling
- **Meet all SLS performance requirements**
- Conduct Network Reliability analysis
- Assess the feasibility of continued use of an ACK based protocol for reliable message delivery and implement as per panel approval.
- Determine whether the Network Services API should consist of a process rather than a set of Library Calls and implement as per panel approval.
- Provide the capability for supporting multiple senders on a single stream (e.g. System Message Service).
- Replace the static address table and its manual configuration with a dynamic address allocation approach.
- Complete requirements not met in Redstone.

APPENDIX B

SLS Requirements

2.2.1.1.2 The RTPS shall be fault tolerant. Specifically, the system shall provide the capability to recover from subsystem failures in the following areas:

4. Real Time Critical Network and the Display and Control Network

2.2.1.1.3 The CLCS shall be designed to have a high level of data integrity.

2.2.1.1.4 The RTPS shall provide fault tolerance in the Command and Control HCI positions. These positions will be connected to the DCN such that a DCN failure will still allow approximately half of the workstations to remain operational. A workstation or network adapter failure will require another workstation to be used.

2.2.1.1.5 The loss of any RTPS Real Time Network component shall not cause switchover of more than one standby subsystem.

2.2.2.1.1 The system shall support 25,000 End-Item Function Designator changes per second continuously. This is the "system maximum data bandwidth".

2.2.2.1.2 The system shall support a peak of 50,000 End-Item Function Designator changes in a given second without losing any data.

2.2.2.1.3 The system shall support 1,000 End-Item Function Designator changes during a 10 millisecond period.

2.2.2.1.8 GSE command/response latency of a priority command, or of a non-priority command in an unloaded system, shall be less than 20 milliseconds from the time a test/control application issues the command until the response is received by the test application.

2.2.2.1.9 GSE command/response latency of a non-priority command in a system supporting the "system maximum data bandwidth" and with 20 test applications executing in the same subsystem shall be less than 100 milliseconds from the time the test/control application issues the command until the response is received by the test application.

2.2.2.1.14 The system shall support executing a manual command in less than one second from a human execution to RTPS interface output.

2.2.2.1.21 The system shall support (while executing at 40 percent of the system maximum data bandwidth) 25 Constraint Management requests from each of 4 Command and Control Processor Subsystems (100 total), 20 constraint event notifications (5 per CCP), and 4 commands (1 per CCP) with each CCP executing 20 User Test Applications which are executing 500 Application Service calls per second.

2.2.3.1.5 The RTPS shall provide changed measurement data to system and user applications at the System Synchronous Rate.

2.2.3.1.6 The RTPS shall provide changed measurement data to display applications at the Display Synchronous Rate.